ISSN 1330-3651 (Print), ISSN 1848-6339 (Online) https://doi.org/10.17559/TV-20240902001963
Original scientific paper

A Novel Consensus Mechanism Using Publicly Verifiable Randomness for Blockchain Networks

S. S. SARANYA*, R. SIVARAJ, M. VIJAYAKUMAR

Abstract: In this paper, we propose a novel consensus mechanism called the Blockchain-based Publicly Verifiable Randomness Algorithm (BCPVRNG-SC) that enhances security, fairness, and transparency in blockchain networks. The mechanism employs two distinct pseudorandom number generators: the Encapsulated Range-Constrained Random Number Generation Algorithm (ER-CRNG) and the Blockchain-based Publicly Verifiable Random Number Generation Algorithm with Smart Contract Integration (BCPVRNG-SG). These algorithms integrate random seeds within the blockchain data structure, ensuring they are publicly verifiable, unpredictable, tamper-resistant, and confidential. The BCPVRNG-SC mechanism promotes equal participation in the consensus process by introducing randomness through entropy sources such as block hashes, timestamps, and external sources like smart contracts. This ensures a fair and unbiased selection process for new block proposers and verifier committees in each round. The use of verifiable random numbers further enhances the integrity and impartiality of the consensus mechanism. To validate the robustness of the newly developed random number generators, we conducted rigorous testing using the NIST SP 800-22 statistical test suite. The results demonstrated excellent randomness properties, with all sequences passing the tests for statistical properties such as correlation, key sensitivity, and uniformity. The findings confirm that the proposed BCPVRNG-SC mechanism meets the desired criteria for security, fairness, and randomness, making it a viable solution for enhancing the efficiency and equity of blockchain consensus algorithms. By employing innovative randomness generation techniques and integrating them seamlessly with blockchain data structures, the BCPVRNG-SC offers a significant advancement in consensus mechanisms, paving the way for more secure and trustworthy decentralized networks.

Keywords: blockchain consensus mechanism; entropy sources; publicly verifiable randomness; random number generation; smart contract integration

1 INTRODUCTION

Random numbers play a pervasive role in everyday life, appearing in various contexts like gambling, welfare lotteries, and more. Moreover, with the expanding openness of the internet, the generation of random numbers has emerged as a crucial focus in the field of computer science. Nakamoto established the direct online payments without need for financial banks to create a decentralized electronic cash system. It solves the problem of double-spending by employing hash-based proof-of-work chains and digital signatures on a peer-to-peer network. A minimal amount of network structure is necessary to check transaction sequences and thwart attacks, as the longest chain with the greatest CPU power permits it. The longest proof-of-work chain is recognized as the official record of events, and nodes are free to join and exit at any time. Bitcoin employs a proof-of-work (PoW) consensus algorithm to achieve agreement on generating new blocks within a specific timeframe [1].

Transparency and provenance become intrinsic features, providing the assurance that transactions are conducted with the knowledge that all involved parties possess the capacity to engage in them. The immutability of records further solidifies the technology's integrity, as information is permanently inscribed and stored without the possibility of modification. Consequently, numerous alternative consensus algorithms have been introduced to address the diverse needs of various applications [2-4].

Certain alternatives [5-7] including proof-of-stake (PoS) and proof-of-importance, raise concerns about the potential skewing of the consensus mechanism in favor of a minority of stakeholders in the blockchain network. As the Proof of Work (PoW) protocol [6, 8], employed by blockchain systems, raises concerns due to its significant resource consumption and tendencies towards centralization of computing resources, the Proof of Stake (PoS) protocol has gained widespread attention as an alternative. In the PoS protocol, a representative node is selected at the start of each round to propose a new block, contingent on the verification of the packing condition by the PoS system. The representative, after validating the longest valid blockchain, proposes a new pending block, broadcasts the newly generated blockchain, and awaits confirmation. Subsequently, in the next round, the PoW system reselects the representative to affirm the results of the prior round. Honest representatives persist in building upon the longest valid blockchain. Conversely, proof-of-elapsed time, proof-of-luck, and proof-of-responsibility necessitate intervention from a centralized authority, contradicting the fundamental principles of a peer-to-peer transaction system [9].

Permissionless blockchain consensus can be fundamentally characterized as a challenge pertaining to the establishment of publicly verifiable randomness. Specifically, the consensus algorithm within a permissionless blockchain system serves as a pseudorandom number generator for the purpose of selecting new block proposers. The development of systems capable of generating publicly verifiable random numbers is a critical aspect of blockchain architecture, as elucidated in [10, 11]. To protect personal privacy from theft or unauthorized access [12, 13] it is essential to guarantee the strong security of random numbers linked to important identifying information.

The rapid creation of random numbers is made possible by the combination of modern technology and the Internet, but there are also potential risks. Even if the Internet has many advantages, there are drawbacks as well, such as the possibility of money or information theft by unscrupulous individuals. Thus, to protect the data of authentic users, random numbers generated over the network need to be extremely secure. A block hash associated with a group of transactions within a block [14, 15].

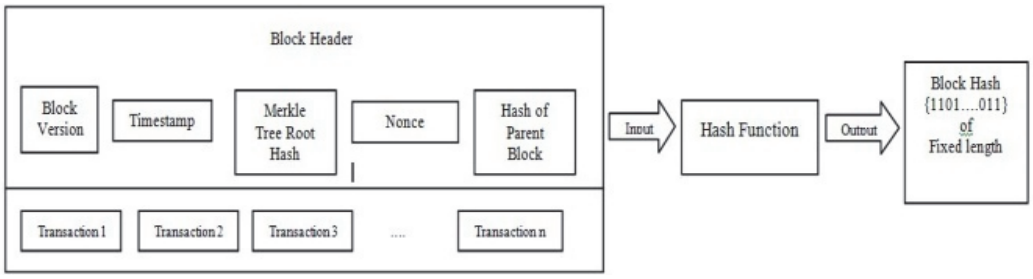

Figure 1 Blockchain hash generation

2 RELATED WORK

A proposed a consensus algorithm by combining Proof-of-Stake (PoS) and Byzantine Agreement protocol (BA). The BA protocol reaches consensus among users for the set of next transactions. For scalability, a new novel technique Verifiable Random Function (VRF) has evolved that permits participants to verify their eligibility to take part in the next transactions. The selection of the users must register in the network via verifiable selection messages between users. With the exception of private key generation, the users must get consensus from other participants for the state of activity that mitigates chosen attacks as it changes the participants after receiving the message. Blockchain architecture serves as an immutable distributed ledger that allows transactions in a decentralized way [16]. To make a consensus among untrusted parties without Byzantine Generals Problem (BG) leads to reaching consensus in a distributed environment.

Of the two categories of consensus algorithms, proof-based and voting-based, the former requires network nodes to prove their eligibility to append transaction blocks to the blockchain. The latter adopts a voting mechanism whereby nodes exchange the results of block verification before appending them to the blockchain. In a voting-based consensus algorithm [17], all nodes participating in the verification process must be known and adaptable to facilitate early message exchange. These nodes must communicate with one another to verify each transaction in a block, thereby determining whether the block should be appended or not.

In a distributed system, a majority of nodes (denoted as T) must hold an identical copy of the proposed block to prevent it from being accepted from a faulty or malicious node. Voting-based mechanisms are designed to resist failures or abnormal behavior from some nodes that could lead to inaccurate results. For fault tolerance, at least t out of N participating nodes must operate correctly. In crash fault-tolerant consensus algorithms, the minimum number of honest nodes required is $(N/2 + 1)$, whereas for Byzantine fault-tolerant consensus, it is $(2N/3 + 1)$.

Consensus algorithms address two major problems, distributed consistency and blockchain integrity, to ensure the efficiency and security of blockchain systems [18]. The Byzantine Generals Problem illustrates the challenge of achieving consensus among honest nodes in the presence of malicious ones. In this context, trustworthy systems must account for malfunctioning components that send conflicting messages to various parts of the system. A failed node may exhibit erratic behavior, often unnoticed, and may send inconsistent information to different nodes.

The Byzantine problem involves two key components: (1) loyal generals must agree on a common course of action, and (2) traitorous generals aim to prevent this agreement. Each general communicates via messages; if $v(i)$ represents a message from the i-th loyal node, then the combined message from all n nodes ($v(1)$, $v(2)$, ..., $v(n)$) is used to make a majority-based decision. A DSS-PP (Data Sharing Security with Privacy Protection) model for data sharing in blockchain-enabled IIoT (Industrial Internet of Things) ensures privacy preservation. To prevent leakage of private device data, identity-based authentication is employed. This not only authenticates user identity but also prevents exposure of sensitive personal information. To optimize storage, only indexed information of sensitive data is stored on the blockchain in encrypted form for public reference.

A two-stage privacy protection scheme is used in mobile crowdsourcing based on blockchain. Emerging edge computing technologies support low-latency data processing at the network edge, followed by broadcasting the processed data to third-party platforms. To enhance the trustworthiness of these platforms, a transparent blockchain-based Double Disturbance Localized Differential Privacy (DDLDP) algorithm is proposed. Blockchain technology not only guarantees data integrity but also prevents third-party platforms from leaking employees' confidential information.

3 PROPOSED WORK

We proposed a consensus algorithm Blockchain based publically verifiable Randomness algorithm (BCPVRNG-SG) that generates unpredictable random numbers for the recipients of blockchain to involve in consensus process. The consensus algorithm arbitrarily chooses a block proposer among list of participants in blockchain. Fundamentally, pseudo random number generators are always deterministic in nature which produces random numbers based on the initial seed value. The vital properties that every random number generator should possess are as listed here; first the sequence should pass all statistical tests to resist different attacks. Secondly, the sequence pattern should be revealed by an attacker to predict the antecedent or descendant from a series of output that are well-known. Third, the random number generator must be a one way function to avoid the problem of guessing the input seed value from the generated output. Random numbers can achieve randomness in the mining process of electing a leader node among participants. The pseudorandom numbers are generated by predetermined algorithms that are predictable and ascertainable when the initial seed value has been revealed. The Publically Provable Random Numbers uses cryptographic hashing

properties to produce publicly provable random numbers in a peer-to-peer environment by appending seed values to data. The proposed algorithm integrates seed value to all generated transactions before it is added to the blockchain to enhance security. The owner has to determine the revealing period of the seed. With fundamental reference to [19] , a new random number generator was proposed which is a publicly verifiable protocol that generates random numbers for each node. The randomness and authenticity of the random numbers of a node have never been revealed to others. The remaining blockchain nodes have no way of knowing the authenticity of random numbers, which leads other nodes to conjecture that it was counterfeit. To circumvent this, nodes have to affix all communicational data to the transactions that have to be verified by other nodes for proving the resulting random numbers were not forged by anyone. The communication parties must sign the resultant communication data with their private key called a digital signature. The digital signature is a stream of random bits, and can only be generated by the owner of the message. The messages are immutable as they are signed with a private key that is only known to the sender. A digital signature is defined as a mathematical procedure that provides the recipients a strong belief that the message originated from a known sender and was not modified or reconstructed during transit. The basic symbols are shown in Tab. 1.

Table 1 Related symbols

Symbol	Description
δ_{start}	Minimum value of random sequence to be generated
δ_{end}	Maximum value of random sequence to be generated
T	Target numbers accuracy
D	Expected Distribution Type
ω	Cumulative sum of ASCII value of all characters in CA_{Str}
τ	Number of all random numbers within a range of δ_{start} & δ_{end} of precision γ
ξ_n	Generated Target random sequence
CA_{Str}	Character Array that stores each character from U_{Str} from right to left using push operation
U_{Str}	String password generated by a Smart Contract
φ_{Seed}	Initial random seed value given as input to Smart Contract
RC_i	Rolling count of a 26 - sided character die to generate a targeted seed string
DX_{26}	A 26 - sided die with each side representing character ranges from *a* to *z*
T_{Sys}	A system date and time
C_i	A character generated by rolling a die during i^{th} iteration
N	Total number of times DX_{26} to be rolled and total number of samples
S	String obtained by rolling a die for 100 times
S_0	Hash of φ_{Seed} , ω and T_{Sys}
S_0 '	A 32-bit of unsigned integer obtained from S_0
S_{final}	100 bit streams containing 3125 S_0 ' seeds

Random Number Generator Algorithm

Algorithm 1: Encapsulated Range-Constrained Random Number Generation Algorithm (ER-CRNG)

Input : δ_{start} , δ_{end} , γ, D

δ_{start} , δ_{end} *range of random number to be generated*

γ, D *limits of random number generated*

Initiate random number function

Generate random sequence is $\xi_1, \xi_2, \xi_3, \xi_4$ …

Obtain Smart contract generated password string U_{Str} *after accepting the generated random sequence* $\xi_1, \xi_2, \xi_3, \xi_4$ ……*as an input*

Create an empty character array CA_{Str}

Iterate through each character in U_{Str} *from last index to the first character*

Store retrieved characters from U_{Str} *string array into* CA_{Str} *character array*

Set $\omega = 0$, *index* ¡ $= 0$ *to iterate through characters from* CA_{Str}

Convert character into ASCII code value num

Calculate $\omega = (num * 256^{\text{i}})$ *and increment* ¡ *by 1 for next iteration*

Calculate $\tau = \omega \,\%\, D$

Calculate $range = (\delta_{end} - \delta_{start})$

Generated random number $\xi = \delta_{start} + (range * (\tau/\omega))$

For $\xi_1, \xi_2, \xi_3, \xi_4$, *find* $\xi_{\text{i}} = (\omega \,\%\, \tau)$

if $\xi_{\text{i}} = (\omega \,\%\, \tau)$*satisfies*

Output: $\xi_1, \xi_2, \xi_3, \xi_4$ ….

else Recalculate $\xi_{\text{i}} = (\omega \,\%\, \tau)$

end

Algorithm 2: Blockchain based Publically Verifiable Random Number Generation Algorithm with Smart Contract Integration (BCPVRNG-SC)

Input: RC_i , DX_{26}, T_{Sys} , N

Initialize $i = 0$, $N = 100$ *where* RC_0 *denotes initial rolling index*

Iteration

Roll 26 sided die with each side representing character ranges from a to z until RC_{100}

Obtain a character for each round of RC_i *based on* i *value*

Concatenate generated characters ξ_i *to String* S *and increment* i *by String* $S = C_1 \| C_2 \| C_3 \| \ldots \ldots C_{100}$

End

Store the targeted string S *as initial seed of random number generator* $\varphi_{Seed} = S$

Obtain Smart contract generated password string U_{Str} *after accepting* φ_{Seed} *as input*

Create an empty character array CA_{Str}

Iterate through each character in U_{Str} *from last index to the first character*

Store retrieved characters from U_{Str} *string array into* CA_{Str} *character array*

Set $\omega = 0$, *index* ¡ $= 0$ *to iterate through characters from* CA_{Str}

Convert character into ASCII code value num

Calculate $\omega = (num * 256^{i})$*and increment* i *by* 1 *for next iteration*

Calculate hash of the obtained string φ_{Seed}, ω *and current system time* T_{Sys} *as timestamp* $Hash = (S_0) = SHA_{256}(\varphi_{Seed}, \omega, T_{Sys})$

Convert the hash value S_0 into 32-bit of unsigned integer to obtain S_0'

Repeat step from 2 *to* 18 *to generate 3125 seeds for 100 times*

Obtain $S_{final} = Concatenate(S_0', S_1', S_2', S_3' \ldots\ldots S_{3124}')$

100 bit streams containing 3125 S_0'seeds

Output: 100 000 bits random sequence.

The consensus algorithm has five different stages. In the first stage, every node generates public and private keys P_{key}^{r-d} and S_{key}^{r-d} for each round. In the second stage, If a node is elected as the block promoter for a particular round through the leader election process, it must generate a new commit message and package the block's transactions for broadcasting to other participants over the network. In the third stage, nodes serving as electors of the proposed block form a committee to verify the block and cast their votes. Once the voting process is completed, the voting results must be propagated to all participants as a broadcast message.In the fourth stage, the node will act as a confirmer of the block through an election process by broadcasting the confirmation message of the currently proposed block to other nodes. In the fifth stage, if any one node receives adequate confirmation messages, it has rights over a block to upload it to a blockchain and terminate the consensus mechanism. The notations of the algorithm are shown in Tab. 2.

Table 2 Notations

Notations	Definition
P_{key}^{r-d} , S_{key}^{r-d}	Public key and private key set of every round *r*
H	A 256-bit hash function
$Block^r$	Block of the current round *r*
$Block^{r-1}$	Block generated in the previous round $r-1$
RN_i^{r-d}	The Random numbers of node *i* are generated for participating in the consensus process of round *r* in step number *d*
Tx_i^r :	Transaction of each block which is generated by the node *i*.
$SIGN_i^{Tx_i^r}$	Digital signature of node *i* to the transaction Tx_i^r .
$Seed^r$	The Seed value generated by leader of the *r*th round, $Seed^r \triangleq H(SIGN_i^r)\ (Seed^{r-1}),\ r)$
$Leader^{r,1}$	Eligible leaders list of 1st step of round *r*
$verifier^{r,d}$	Eligible block verifiers list of the round *r* of the 2nd step
$Confirmer^{r,d}$	Eligible block confirmers of the round *r* of the 3rd step

Key Generation

The public key cryptosystems have five specific components to produce public and private key pairs. The following are the fundamental services provided by public-key cryptography.

Algorithm 3: Key Generation

Key Generation - Creates key pairs P_{key} and S_{key}

Signature Creation - The signature of a message is created using the sender's private key to ensure the authenticity of the message. This signature can only be generated by the rightful owner of the transaction. The signature is defined as:

$SIGNATURE = SIGN(T_x, S\_\text{key})$

Signature Verification - Upon receiving a transaction from the proposer, the message's authenticity is verified using the sender's public key, which has already been shared with all other participants in the consensus mechanism. Signature verification ensures that the transaction truly originates from the legitimate owner and not from an unauthorized source:

$Verify(T_x, SIGN(T_x, S\_\text{key}))$

Each node $i \in N$ in the consensus algorithm generates a unique set of public and private keys for each round of the key generation process. These keys are denoted as: $P\_\text{key}\hat{}(r-d)$ and $S\_\text{key}\hat{}(r-d)$ where r represents the round number and d represents the step within each round. Tab. 3 outlines the key generation process and provides the notation used for each operation.

Table 3 Process of key generation and its notations

Functionality	Enlightenment
Key Generation	Generate both public and private keys for the use of other functionalities in each round. The protocol promotes key exchanging functionalities among the parties for further communication.
Process of Encryption	Transaction Encrypted, $E(T_x) = Encrypt(T_x, P_{\text{key}})$
Process of Decryption	Transaction Decrypted $D(T_x) = Decrypt(E(T_x), S_{\text{key}})$
Signature of the Transaction	$SIGNATURE, SING(T_x S_{\text{key}})$
Verification	Verification = $\text{Verify}(T_x, SING(T_x S_{\text{key}}))$

Transaction Creation

The random number generated by the owner of the transaction is known as the $Owner_{\text{seed}}$. Instead of transferring the $Owner_{\text{seed}}$ to other participating nodes directly, this must be reconstructed by a hashing algorithm. The hashed value can be appended as a part of transactional data. The algorithm creates a final hash of the transaction TxH_{owner} before being appended to the blockchain. As the transaction creation is performed over a constant period, the complexity of the transaction creation process is linear, $O(n)$, where n is the number of transactions appended to the block. During transaction proposal phase, the leader of previous round must reveal its previously generated secretes value $Seed_{\text{Prev}}$ and commitment Commit $Seed_{\text{Prev}}$. For every round the leader has to propose a new data set DS_r for the upcoming round *r* of step *d*. The data set created by the block proposer must include the following data into to block header.

The dataset consists of all the transactions $T_{x1}, T_{x2}, \ldots. T_{xn}$

The hash value of the body (i.e) hash of the transactions Hash (DS_r)

The current index of round *r* and step *d*.

The current rounds specific random numbers RN_r.

The seed of the leader of previous round $Seed_{\text{leader}}$.

The previous round $r-1$, its corresponding dataset DS_{r-1}

Hash of previous data set value DS_{r-1} is $Has-\left(DS_{r-1}\right)$

The hash of the Merkle Tree Root (MTR_r) is computed over the sum of all encrypted transactions generated during the commitment phase of the new leader, based on the cumulative shares of all transactions conducted prior to the leader's election.

Algorithm 4: Transaction Creation Each node $i \in P_$key in round *r* and step *d* initiates step *d* of round *r* immediately after the completion of the previous block, denoted as Block^(r−1).

The random number RN^r is calculated by using the previously generated block $Block^{r-1}$. $RN^r \triangleq H(SIGN_I\left(RN^{r-1}\right), r)$.

The random number and transaction of node i was combined and hashed together to create a transaction message. $Block^{r-1} \triangleq H(SIGN_I(RN^r, Tx_i), S_{\text{key}})$.

Block Proposal

During the block proposal phase of the consensus process, a leader for the current round must be elected by the other participants. Any node wishing to become the leader must reveal the previously committed seed value **S_{r−1}** and generate a new commitment. Once the previously committed secret is revealed, the other participants must validate it and issue a confirmation certificate **CC_i(DS_{r−1})** to acknowledge that **DS_{r−1}** was a valid dataset from the previous round.

Algorithm 5: Block Proposal

The node *i* reveal the secret value S_{r-1} of previously generated round *r*. The remaining participants of consensus process must acknowledge the recent commitment by producing the confirmation certificate for the dataset DS_{r-1} of the previous round.

Node i utilizes the secrete seed value of previous round S_{r-1} to computing the Hash value of $H\left(SIG_i\left(S_{r-1}, r\right)\right)$ based on the $block_{r-1}$.

Node i receives confirmation $CC_i\left(DS_{r-1}\right)$ from participants by $CC_i^{r,1} \triangleq SIGN_i\left(r, 1, Seed_{r-1}, RN_i, Tx_i\right)$

The signature $CC_i^{r,1}$ has sent to the hashing algorithm to generate $H\left(SIG_i\left(r, 1, S_{r-1}, RN_i, T_x\right)\right)$ where node *i* belongs to the of public key set $i \in P_{\text{key}}^{r-d}$ which is also equivalent to $Leader_{r,1}$ .There are two scenarios in this block proposal process which are as follows,

if the node *i* is not an element of the potential leader set $Leader_{r-1}, i \in P_{\text{key}}^{r-d}$ if so, stop its execution.

If the node *i* belongs to the potential leader set $Leader_{r-1}$, $i \in P_{\text{key}}^{r-d}$ of round *r*, then node *i* computes all the transactions generated in the round *r*. The final block of round *r*, $block_i^r$.

The $H\left(block^{r-1}\right)$ is the Merkle Tree Root MTR_r which is computed by integrating hash of the Merkel Tree Root with all the previous transactions and transactions generated by node *i*, Tx_i^r . The hash of Merkle Tree Root MTR_r is $H\left(block^{r-1}, Tx_i^r\right)$, computed by combining the entropy of every transaction in a block. Tx_i^r differentiates the transaction owner node i from other participants since the block proposer does not aware of the seed value generated by the owner, not data can be manipulated in the block as final hash value given to owner and proposer are not same. The Merkle Tree Root MTR_r is publically verifiable in the blockchain.

The block $block_i^r = \left(r, Tx_i^r SIGN_i^{r-1}, H\left(block^{r-1}\right)\right)$, $SIGN_i^{r-1}, H\left(block^{r-1}\right)$ is defined as a candidate block of the current round.

Finally, the message of node i in rth round, computes $Message_i^{r,1} \triangleq (block_i^r, SIGN_i\left(H\left(block_i^r\right), CC_i^{r,1}\right)$ and propagate to other participants.

In step 1 of the consensus process, the potential leader has been chosen, node *i* was chosen as a leader of the round r that was removed from participating in the next step of the consensus process which enhances the security of the consensus protocol. After removing the node *i* have to wait for $f+1$ round. To ensure that the correct node has been chosen as a leader, for the next $f+1$ rounds the potential leader list. The previously elected potential leaders list chosen for the last *f* rounds must be removed from the list, which may not be eligible for the current round election process. The committed data set must be signed by the private key of a single honest leader of the same round. Each node receives the message $Message_i^{r,1} \triangleq (block_i^r, SIGN_i\left(H\left(block_i^r\right), CC_i^{r,1}\right)$ from the leader of the current node at the end of the current proposal phase. The upcoming round leader firstly verified the created dataset DS_{r-1} where it was delineated and signed properly.

Voting Process

Every node $i \in P_{\text{key}}^{r-d}$ and $\notin leader^{r-1}$, node i starts step 2 for voting in round r as soon as the previously generated $block^{r-1}$.

Algorithm 6: Voting a Block

The secrete seed value of previous node is calculated as $Seed_{r-1} \triangleq H\left(SIGN_i\left(Seed_{r-2}, r\right)\right)$ using the previous block $block^{r-1}$.

The node i has to resolve whether it belongs to the verification committee list, $verifier_{r,2}$ of step 2 in round *r*.

$CC_i^{r,2} \triangleq SIGN_i\left(r, 2, Seed_{r-1}, RN_i, Tx_i\right)$ must be calculated, and it should satisfy the following scenario $H(CC_i^{r,2}) \triangleq H\left(SIGN_i\left(r, 2, Seed_{r-1}, RN_i, Tx_i\right)\right)$ equivalent to $Verifier_{r,2} \triangleq \{i \in P_{key}^{r-d}, H(CC_i^{r,2})\}$

There are two scenarios in this voting process which are as follows,

If the node i is not an element of the potential verifier set $Verifier_{r,2}$, $i \notin P_{\text{key}}^{r-d}$ it stops execution.

If the node i belongs to the potential verifier set $Verifier_{r,2}$, $i \in P_{\text{key}}^{r-d}$ of round r, then node i computes all the transactions generated in the round r. The final block of round r, $block_i^r$.

The node i must verify all the messages $Message_i^{r,1}$ received from potential leaders using the valid credential $CC_i^{r,2}$.

The node i checks the validity of the random number RN_i for the message Tx_l .

The node i also validates the other credentials like $CC_i^{r,1}$, if legal or not.

The node finds $H\left(block_i^r\right)$ from the message $Message_i^{r,1}$ and computes the confirmation message for voting from all other participants. $Vote_i^r \triangleq H\left(block_i^r\right)$.

It discarded the credential $CC_i^{r,1}$ and restarted from the step 4. Finally the vote message is generated $Message_i^{r,2} \triangleq (SIGN_i\left(i, Vote_i^r, SIGN_i\left(Vote_i^r\right), CC_i^{r,2}\right)$ and propagated to all other participants in the consensus process.

Block Confirmation

Every node $\in P_{\text{key}}^{r-d}$, $i \notin leader^{r,1}$ and $i \notin verfier^{r,2}$ node i starts step 3 for voting in round r as soon as the previously generated $block^{r-1}$

Algorithm 7: Block Confirmation

The node i reveals the secret value $Seed_{r-1}$ of previously generated round r. The node i has to resolve whether it belongs to the verification committee list, $\text{confirmer}_{\text{r,3}}$ of step 3 in round r.

$CC_i^{r,3} \triangleq SIGN_i\left(r, 3, Seed_{r-1}, RN_i, Tx_i\right)$ must be calculated, and it should satisfy the following scenario $H(CC_i^{r,3}) \triangleq H\left(SIGN_i\left(r, 2, Seed_{r-1}, RN_i, Tx_i\right)\right)$ equivalent to $Confirmer_{r,3} \triangleq \{i \in P_{key}^{r-d}, H(CC_i^{r,3})\}$.

Blockchain Creation

Every node $i \in P_{\text{key}}^{r-d}$ starts step 4 for uploading the block in round r as soon as the previously generated $block^{r-1}$ is known.

Algorithm 8: Block Creation

The node i must verify the confirmation messages that are gathered from other participants $Message_i^{r,3} \triangleq (SIGN_i(i, FinalConfirmation_i^r, SIGN_i(FinalConfirmation_i^r), CC_i^{r,3})$

There are two scenarios in this blockchain creation process which are as follows,

If the node i receives sufficient block confirmation messages

the block will be finally added to the blockchain. Otherwise, the block $block^r$ will not be added to the blockchain.

4 SIMULATION RESULTS

The proposed random number generator is tested using the statistical NIST SP 800-22. To test the randomness of the proposed RNG, we have generated 1000 sequences of 106 bits length each. There are 15 categories of test that has been used to analyze the generated sequence. It is considered that p-value which is larger than α passed the statistical test. For consideration, the range of α is said to be 0.001 to 0.01. The p-value of each test is compared with α, if it is greater, that will be affirmed as random, otherwise not.

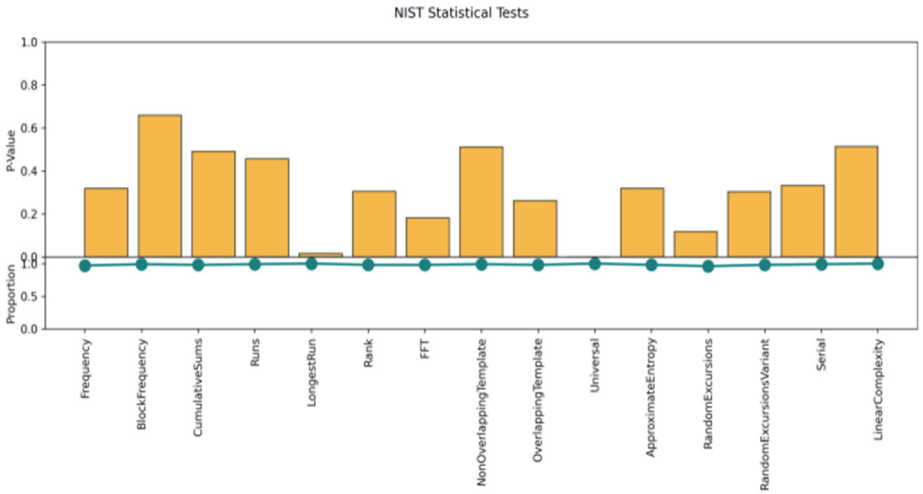

Figure 2 Plot of the statistical test suite (NIST SP -200)

For simulation, we set α is 0.01 and the results of our proposed PRNG trialing shown in Tab. 4, substantiated our primary perception and affirmed sufficient proof that the sequence of random number generated as a result of the proposed technique is calculated to be random as well. Even though passing all statistical tests does not assure

randomness, conversely it may be acknowledged that no evident patterns were encountered while analyzing the generated data. The minimum pass rate is approximately = 96 for a sample size = 100 binary sequences. The minimum pass rate for the random excursion (variant) test is approximately = 12 for a sample size = 14 binary sequences. Fig. 2 shows the results of the Statistical Test Suite (NIST SP -200) as a plot.

Security Analysis

Kolmogorov - Smirnov Goodness of Fit Test [KS-Test] for Uniformity. The random number sequence generated using proposed algorithm should acquire property of uniformity. If the sequence is defined as u_1,u_2, …, u_N then the uniformity can be tested by inspecting independent distribution of numbers between ones and zeros.

Security Analysis

Kolmogorov - Smirnov Goodness of Fit Test [KS-Test] for Uniformity. The random number of sequences generated using proposed algorithm should acquire property of uniformity. If the sequence is defined as u_1, u_2, …, u_N then the uniformity can be tested by inspecting independent distribution of number between ones and zeros. If the distribution is said to be uniform, the generated random sequences are actually independent. Hypotheses test can be done to predict the uniformity of the sequence.

$H_0 - u_i$ has balanced zeros and ones, where as

$i = 1, 2, 3, \ldots..N$

$H\alpha - u_i$ - has no balanced zeros and ones, where as

$i = 1, 2, 3, \ldots..N$

In null hypothesis (H_0) the numbers are definitely uniform whereas another hypothesis (H_α) affirms that the sequence is not uniform. If the p-value returned from the test is greater than the critical value (α) we could probably reject the null hypothesis (H_0), by believing that the sequence does not have a uniformly distributed zeros and ones. The Empirical Cumulative Distribution Function (ECDF) is denoted as Fn calculated for N number sequence. The 32 bit random sequence generated must have 16 ones and 16 zeros. Due to unbalanced zeros and ones, the generated random sequence does not possess uniformity. Slight variations of ratio of zeros and ones may be considered as uniform.

$$Fn(t) = \frac{numbers\ in\ the\ sequence \le t}{N}$$

The following equation Eq. (2) depicts the equilibrium degree (ϵ) for the given sequence of N bits length.

$$\epsilon = \frac{|C_0 - C_1|}{N}$$

In Eq. (2), N is defined the total number of bits in the sequence with C_0 number of Zero's count and C_1 number of one's count. The equilibrium ϵ is equal to zero when the sequence has the equal number of zeros and ones. The Fig. 3 shows the distribution of zeros and ones and ones count of the sequence N and Tab. 4 shows the results of Kolmogorov-Smirnov (KS) goodness of fit test for each sample for uniform distribution. The minimum pass rate of the Kolmogorov-Smirnov (KS) goodness of fit test for uniform distribution is 95/100 , number of rejections may be permitted up to 5 (Rejections < 5).

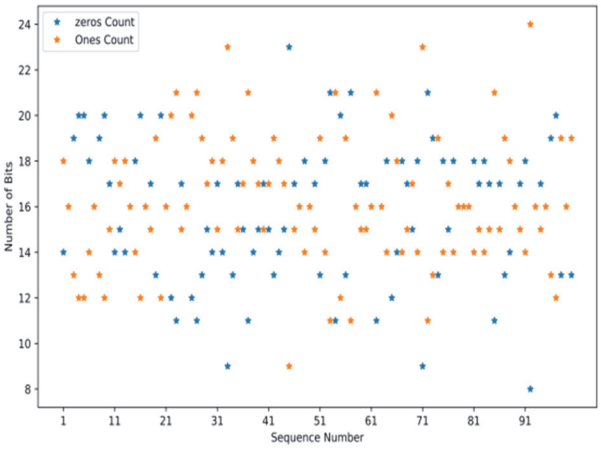

Figure 3 Distribution of zeros and one in the sequence

Table 4 Results for Kolmogorov-Smirnov (KS) goodness of fit test (uniform distribution)

Sample number	1
Status of Reject H_0 - values are distributed uniformly (KS α = 0.05)	Accepted Uniform distribution(TRUE)
Number of random numbers	3125
Mean	0.497353388504535
Min	0.000301289837807
Max	0.999519011005759
Variance	0.083408356150500
Standard deviation	0.288805048692886
P value	0.768545388719236
Sample number	2
Status of Reject H_0 - values are distributed uniformly (KS α = 0.05)	Accepted Uniform distribution(TRUE)
Number of random numbers	3125
Mean	0.489711427283287
Min	0.000147282145912
Max	0.999594202730804
Variance	0.081773317644511
Standard deviation	0.285960342782895
p-value	0.165849688377660
.....	
Sample number	100
Status of Reject H_0 - values are distributed uniformly (KS α = 0.05)	Rejected Uniform distribution (FALSE)
Number of random numbers	3125
Mean	0.5035533951576799
Min	0.0008397193159908
Max	0.9999037513043731
Variance	0.0830556712912954
Standard deviation	0.2881938085582260
P value	0.5831131418323743
Summary of samples (n =100)	
Status of Reject H_0 . values (Cumulative KS)	2 out of 100 Samples
Number of runs	100
Number of seeds	3125
Mean standard deviation	0.288734107
Minimum seeds of 100 sample runs	0.000186605
Mean of minimum seeds	0.000370318

In summary, out of 100 samples, 2 observations have been rejected. The rate of Kolmogorov-Smirnov (KS) goodness of fit test is 98/100. The Fig. 4 shows the sample

of minimum value and running mean of the sample sequence whereas Tab. 5 depicts the observation of minimum mean for each sample sequence.

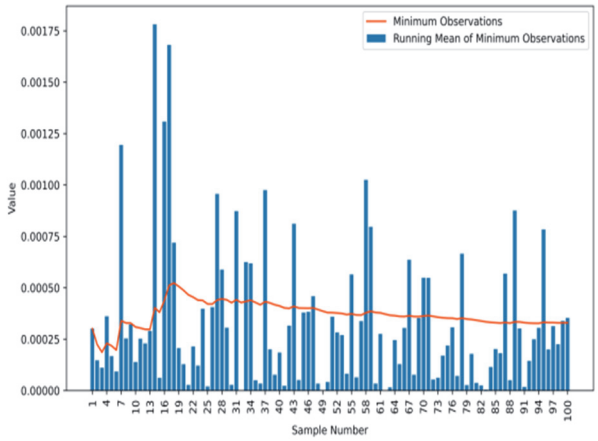

Figure 4 Samples of minimum value and running mean of the sample sequence

Table 5 Observation of minimum and running mean for each sequence

Samples	Minimum	Running mean
1	0.00030129	0.00030129
2	0.00022428	0.00026278
3	0.00037031	0.00022469
	……..	
98	0.00032948	0.000329982
100	0.00032958	0.000329781

For scalability assessment, the simulation conducted with 10,000 nodes demonstrates the approach accomplishes invariable latency by maintaining the transaction throughput. The computational overhead distributed over network nodes mitigates bottlenecks by adopting decentralized verification. Compare to conventional on-chain random number generation oracles, such as Chainlink VRF the proposed BCPVRNG-SC mechanism decreases gas costs not by relying on external oracles. When generating 100,000 bits that is equivalent to 3,125 seeds, traditional RNG solutions on-chain would cost approximately 0.1-0.15 ETH per transaction. In contrast, BCPVRNG-SC is capable of finishing the similar practice at 0.08 ETH, resulting in a 20% savings in gas fees.

5 CONCLUSION

A novel approach to utilizing permissionless blockchain systems as pseudo-random number generators, producing confidential, tamper-resistant, unpredictable, and collision-resistant publicly verifiable random numbers. The proposed scheme ensures randomness by generating verifiable random numbers that successfully pass rigorous tests, including the NIST SP 800-22 statistical test suite. The computation increases transaction latency in resource-constrained environments. Utilizing lightweight cryptographic hash functions computationally efficient while preserving security. Elevated latency management among nodes possibly will affect the timeliness and consistency of random number generation. Robust safeguards must also require monitoring and mitigating the malicious behavior.

The dependence on initial conditions guarantees absolute randomness, but it also introduces a potential risk. Real world application encounters regulation and compliance specifically in financial and government sector while ensuring the integrity and validity of the random numbers generated during disputed periods can be complex.

6 REFERENCES

[1] Yang, N., Tang, C., Zhou, Q., & He, D. (2023). Dynamic Consensus Committee-Based for Secure Data Sharing With Authorized Multi-Receiver Searchable Encryption. *IEEE Transactions on Information Forensics and Security*, *18*, 5186-5199. https://doi.org/10.1109/TIFS.2023.1234567

[2] Fu, H., Nie, W., & Liu, L. (2021). A Ring Signature Trust Model for Project Review Based on Blockchain Smart Contract. *Tehnički vjesnik*, *28*(2), 347-356. https://doi.org/10.17559/TV-20210201

[3] Li, A., Wei, X., & He, Z. (2020). Robust Proof of Stake: A New Consensus Protocol for Sustainable Blockchain Systems. *Sustainability*, *12*(7), 2824. https://doi.org/10.3390/su12072824

[4] Nguyen, C. T., Hoang, D. T., Nguyen, D. N., Niyato, D., Nguyen, H. T., & Dutkiewicz, E. (2019). Proof-of-Stake Consensus Mechanisms for Future Blockchain Networks: Fundamentals, Applications and Opportunities. *IEEE Access*, *7*, 85727-85745. https://doi.org/10.1109/ACCESS.2019.2925142

[5] Xiao, B., Jin, C., Li, Z., Zhu, B., Li, X., & Wang, D. (2021). Proof of Importance: A Consensus Algorithm for Importance Based on Dynamic Authorization. *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, 510-513. https://doi.org/10.1109/WI-IAT51129.2021.00095

[6] Nguyen, G.-T. & Kim, K. (2018). A Survey About Consensus Algorithms Used in Blockchain. *Journal of Information Processing Systems*, *14*(1), 101-132. https://doi.org/10.3745/JIPS.02.0083

[7] Xiong, H., Jin, C., Alazab, M., Yeh, K.-H., Wang, H., Gadekallu, T. R. et al. (2021). On the Design of Blockchain-Based ECDSA with Fault-Tolerant Batch Verification Protocol for Blockchain-Enabled IoMT. *IEEE Journal of Biomedical and Health Informatics*, *26*(5), 1977-1986. https://doi.org/10.1109/JBHI.2021.3065895

[8] Iqbal, M. & Matulevičius, R. (2021). Exploring Sybil and Double-Spending Risks in Blockchain Systems. *IEEE Access*, *9*, 76153-76177. https://doi.org/10.1109/ACCESS.2021.3083390

[9] Fernando, P., Dadallage, K., Gamage, T., Seneviratne, C., Braeken, A., Madanayake, A. et al. (2023). Distributed-Proof-of-Sense: Blockchain Consensus Mechanisms for Detecting Spectrum Access Violations of the Radio Spectrum. *IEEE Transactions on Cognitive Communications and Networking*. https://doi.org/10.1109/TCCN.2023.3147981

[10] Du, Y., Wang, Z., Li, J., Shi, L., Jayakody, D. N. K., Chen, Q. et al. (2022). Blockchain-Aided Edge Computing Market: Smart Contract and Consensus Mechanisms. *IEEE Transactions on Mobile Computing*. https://doi.org/10.1109/TMC.2022.3145987

[11] Mali, M., D'Souza, A., Thombare, V., Taskar, R., & Kamthe, A. (2025). Blockchain-enabled decentralized autonomous organizations (DAOs) for sustainable supply chain management. *Journal of Innovations in Business and Industry*, *3*(2), 115-122. https://doi.org/10.61552/JIBI.2025.02.004

[12] Bezuidenhout, R., Nel, W., & Maritz, J. M. (2022). Defining Decentralisation in Permissionless Blockchain Systems. *The African Journal of Information and Communication*, *2022*(29), 1-24. https://doi.org/10.23962/10539/34013

[13] Liu, Z., Huang, B., Hu, X., Du, P., & Sun, Q. (2023). Blockchain-Based Renewable Energy Trading Using Information Entropy Theory. *IEEE Transactions on Network Science and Engineering*. https://doi.org/10.1109/TNSE.2023.3148765

[14] Bezuidenhout, R., Nel, W., & Maritz, J. M. (2022). Embedding Tamper-Resistant, Publicly Verifiable Random Number Seeds in Permissionless Blockchain Systems. *IEEE Access*, *10*, 39912-39925. https://doi.org/10.1109/ACCESS.2022.3165897

[15] Zhang, P., Yang, P., Kumar, N., Hsu, C.-H., Wu, S., & Zhou, F. (2023). RRV-BC: Random Reputation Voting Mechanism and Blockchain Assisted Access Authentication for Industrial Internet of Things. *IEEE Transactions on Industrial Informatics*. https://doi.org/10.1109/TII.2023.3156784

[16] Zhang, Q., Li, Y., Wang, R., Liu, L., Tan, Y., & Hu, J. (2021). Data Security Sharing Model Based on Privacy Protection for Blockchain-Enabled Industrial Internet of Things. *International Journal of Intelligent Systems*, *36*(1), 94-111. https://doi.org/10.1002/int.22244

[17] Sun, Z., Wang, Y., Cai, Z., Liu, T., Tong, X., & Jiang, N. (2021). A Two-Stage Privacy Protection Mechanism Based on Blockchain in Mobile Crowdsourcing. *International Journal of Intelligent Systems*, *36*(5), 2058-2080. https://doi.org/10.1002/int.22499

[18] Gouget, A., Patarin, J., & Toulemonde, A. (2021). Leader Election Protocol Based on External RNG Services. *2021 3rd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*, 73-80. https://doi.org/10.1109/BRAINS52497.2021.9617389

[19] Bezuidenhout, R., Nel, W., & Maritz, J. M. (2023). Permissionless Blockchain Systems as Pseudo-Random Number Generators for Decentralized Consensus. *IEEE Access*, *11*, 14587-14611. https://doi.org/10.1109/ACCESS.2023.3154987

Contact information:

Mrs. S. S. SARANYA
(Corresponding author)
Department of Computer Science and Engineering,
PSG Institute of Technology and Applied Research,
Coimbatore, Tamilnadu, India
E-mail: saranya.sowndarajan@outlook.com

Dr. R. SIVARAJ
Department of Computer Science and Engineering,
Nandha Engineering College,
Perundurai, India
E-mail: rsivarajcse@gmail.com

Dr. M. VIJAYAKUMAR
Department of Computer Science and Engineering,
Sasurie College of Engineering,
Vijayamangalam, India
E-mail: tovijayakumar@gmail.com